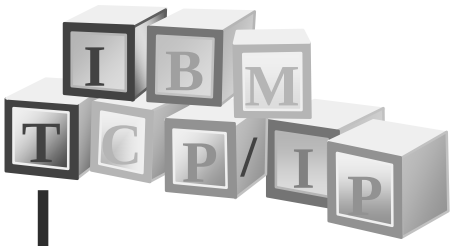
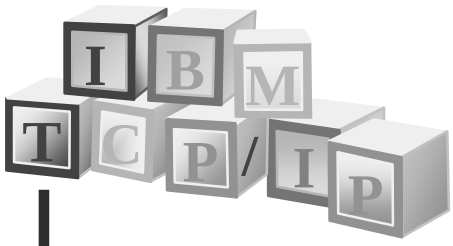


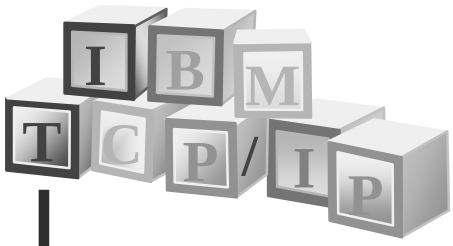
Winsock 1.1 for OS/2

Merideth Norris (merideth@rtpnotes)
Alfred Daun (daun@rtpnotes)
Andre Asselin(asselin@rtpnotes)



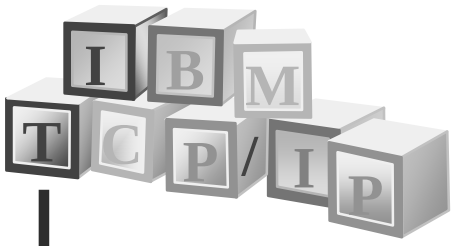
Overview

- Description
  - Provides Winsock 1.1 APIs for native OS/2 socket applications that run in either Open32 or PM environment.
- References
  - Windows Sockets Version 1.1 (www.stardust.com)
  - IBM TCP/IP for OS/2 Warp Programming Reference
  - OS/2 Winsock 1.1: Design & Architecture (os2/design/winsock.sam in CMVC)
  - Winsock 2 Debug and Trace Facilities (dbgspec.doc which can be found in the "Winsock 2 Specifications and Header Files" page provided by www.stardust.com)



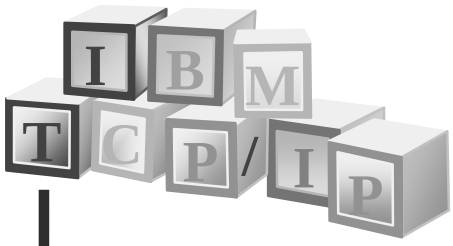
Terminology

- Asynchronous Functions (WSAAsync*) - asynchronous versions of the standard getXbyY() and select() functions.
- Blocking Hook Support - functions called by Windows to check the completion of a blocking socket call.
- DAPWSOCK.DLL/LIB - Open32 version of Winsock 1.1 API library.
- Open32 - A set of Windows APIs used mainly for porting Windows applications to OS/2.
- PMWSOCK.DLL/LIB - PM version of Winsock 1.1 API library.



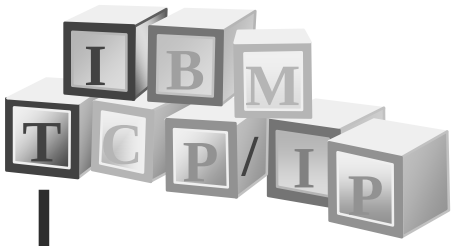
Overview - Features

- Supported
  - All Winsock 1.1 APIs
  - WSTrace/WSFormat (based on Winsock 2 Debug/Trace Document)
- Unsupported
  - NA



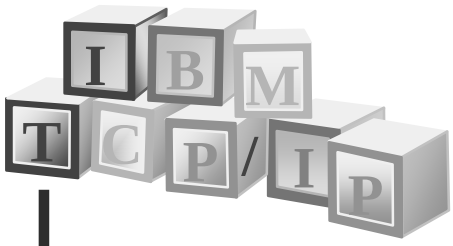
Limitations & Dependencies

- Runs on OS/2 V4.0 (Merlin) or later releases.



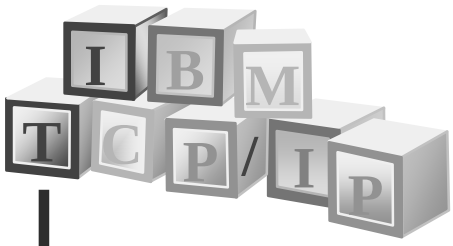
Installation

- Installed files that make up this function
  - \mptn\dll\dapwsock.dll
  - \mptn\dll\pmwsock.dll
  - \tcpip\bin\wstrace.cmd
- Migration/Compatibility Issues
  - NA



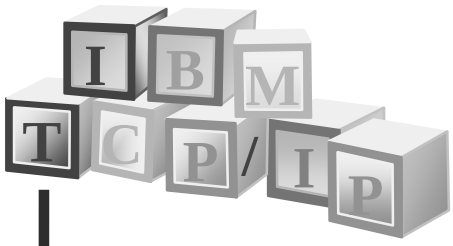
Configuration & Setup

- NA



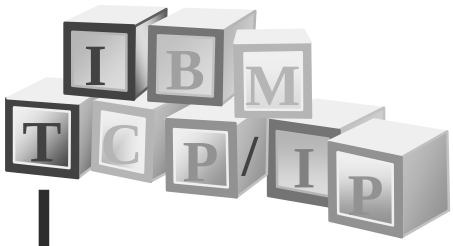
Invocation & Startup

- All Winsock APIs are called by user .exe or .dll. The first Winsock function to be called MUST BE WSAStartup().



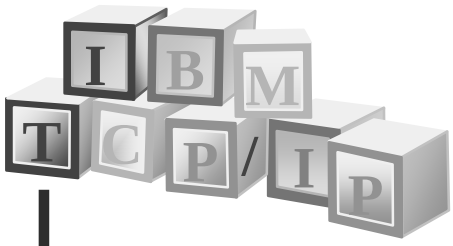
Error Messages, Codes & Logs

- Messages
  - – each API has a set of returned messages, refer to Windows Sockets 1.1 Specification for more detailed information.
- What error logs are available
  - debug trace, see "Debugging and Troubleshooting".



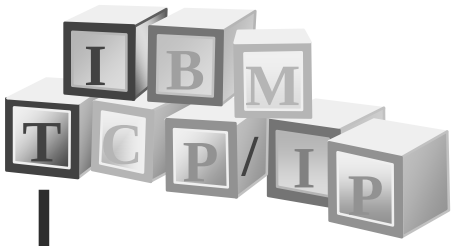
Common User Problems

- Problem: Did not receive any messages from WSAAsync* APIs
  - Cause: When compiling the application, it probably linked to the wrong .LIB.
  - Solution: Make sure the correct .LIB file is linked: DAPWSOCK.LIB for Open32 (Windows APIs) applications; PMWSOCK.LIB for PM applications.
- Problem: Winsock API failed
  - Cause: can be anything from invalid parameters to bad connection.
  - Solution: check the return code from the API and, if available, what is the return value from WSAGetLastError()?



Debugging & Troubleshooting

- Types of information needed
  - Application logs, if available.
  - WSTRACE/WSFORMAT - provides entry/exit information of each WINSOCK function calls.
  - INETDBG - provides entry/exit information of the "real" socket APIs.
  - IPTRACE/IPFORMAT - provides detailed information on sent/received packets.
  - NETSTAT - provides network status of local workstation, e.g., socket connection, network interfaces, etc..
  - Note: Instruction can be found in either the user's guide or programming guide.



WinSock Trace Overview

- WinSock 1.1 for OS/2 provides software developers with a sockets layer trace mechanism.

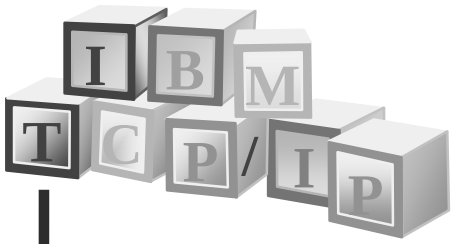
- Each Winsock API call is traced upon entry and just prior to exit from the OS/2 WinSock DLL (DAPWSOCK.DLL or PMWSOCK.DLL)

- The WSTRACE command enables and disables the WinSock trace in an OS/2 Session.

  - WSTRACE is shipped with the base operating system.
  - Trace is recorded in binary format.

- The WSFORMAT command converts the binary trace data into a readable format.

  - WSFORMAT is shipped in the Warp 4.0 Toolkit, which is available on DevCon 11.



WSTRACE
Command Syntax

•wstrace [on | off]
[filename | -p [pipename]] [-b buffsize]

•**Parameter descriptions**

on/off

Turns trace on/off, the default is on.

filename

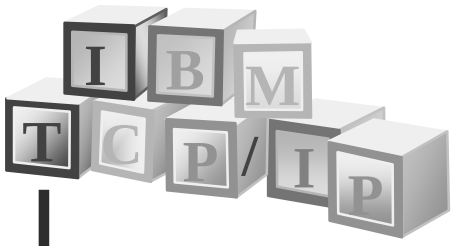
specifies the file name that the trace output will be written to. COM1-COM4 are acceptable filenames.

-p *pipename*

Specifies the name of an OS/2 pipe that the trace output will be written to. If *pipename* is not specified with the -p parameter, the pipename defaults to WSTRACE.

-b *buffsize*

Specifies the size of the global trace buffer. The default is 64K.



WSTRACE Continued

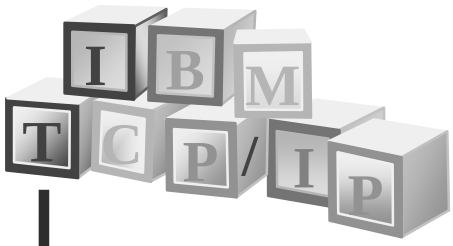
- **WSTRACE Remarks**

  - Tracing must be activated in the same OS/2 session as the WinSock application being traced.

  - When neither *filename* nor *pipename* is specified, the default is to write the trace output to the file, WSTRACE.DMP in the local directory.

  - When tracing to COM Ports, use the OS/2 MODE command to ensure that the output and input COM Port settings are compatible. Otherwise, unexpected results may occur.

  - When specifying COM Ports or pipes as output, WSFORMAT must be invoked and ready to receive the binary trace data as input before executing the application being traced.



WSFORMAT Command Syntax

•wsformat [*input_filename* | [-p *input_pipename*]]
[-f [*formatted_output_filename*]]
[-b [*binary_output_filename*]]

•Parameter descriptions

input_filename

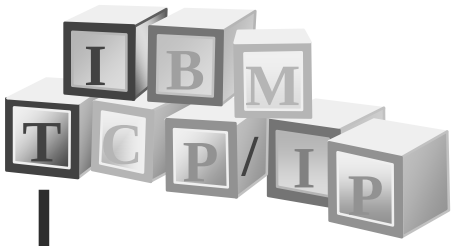
Specifies the name of the input file that contains the trace data to format. COM1-COM4 are acceptable file names.

-p *pipename*

Specifies the name of the OS/2 pipe that contains the trace data to format. If a *pipename* is not specified with the -p parameter, the pipename defaults to WSTRACE.

-f *formatted_output_file*

Specifies the name of the file that the formatted trace output will be written to. If *formatted_output_file* is not specified with the -f parameter, the filename defaults to WSFORMAT.DMP in the local directory.



WSFORMAT Continued

- **Parameter descriptions Con't.**

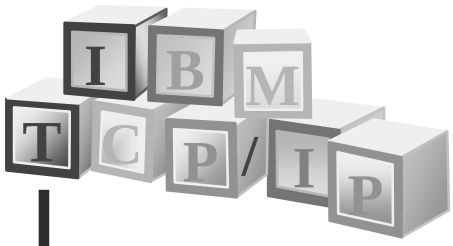
-b *binary_output_file*

Specifies the name of a file that will receive a copy of the binary data being formatted. This can be used to save a copy of the trace information coming in from a COM Port or pipe. If *binary_output_file* is not specified with the -b parameter, the output file name defaults to WSTRACE.DMP in the local

- **WSFORMAT Remarks**

  - If an output source is not specified, the formatted trace output defaults to the screen.

  - When specifying COM Ports, use the OS/2 MODE command to ensure that the input and output ComPort settings are compatible. Otherwise, unexpected results may occur.



Winsock Trace Example

WSFORMAT Version Warp 4.00.00 ***Formatter Version***
Winsock Trace - Version 1 ***WinSock Version***
Trace Date 07/15/1996 Trace Time 15:58:44.25 ***TimeStamp from Trace***

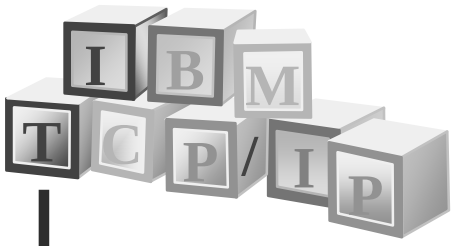
Thread TimeStamp Winsock Function (Parameters) ***Trace Heading***

Thread ID ***WinSock Function Called*** ***Parms passed to and returned from the WinSock Function***

00001 15:58:54.44 WSASTARTUP Exit (wVersionRequired=101, lpWSADATA=48708 Return=0)

Trace Entry TimeStamp ***Function Enry/Exit Indicator***

00001 15:58:54.47 NTOHS Entry (netshort=43981)
00001 15:58:54.50 NTOHS Exit (netshort=43981, return=52651)
00001 15:58:54.53 HTONS Entry (host=52651)
00001 15:58:54.53 HTONS Exit (host=52651, return=43981)
00001 15:58:54.53 NTOHL Entry (netlong=2882409012)
00001 15:58:54.53 NTOHL Exit (netlong=2882409012, return=873647787)
00001 15:58:54.53 HTONL Entry (host=873647787)
00001 15:58:54.53 HTONL Exit (host=873647787, return=2882409012)
00001 15:58:54.53 WSACLEANUP Entry ()
00001 15:58:54.53 WSAISBLOCKING Entry ()
00001 15:58:54.53 WSAISBLOCKING Exit (FALSE)
00001 15:58:54.84 WSACLEANUP Exit (Return=0)
00001 15:59:01.47 WSAISBLOCKING Entry ()
00001 15:59:01.47 WSAISBLOCKING Exit (FALSE)
00001 15:59:01.47 WSASTARTUP Exit (wVersionRequired=101, lpWSADATA=48708 Return=0)
00001 15:59:01.47 GETPROTOBYNUMBER Entry (proto=0)
00001 15:59:01.47 WSAISBLOCKING Entry ()
00001 15:59:01.47 WSAISBLOCKING Exit (FALSE)
00001 15:59:01.50 GETPROTOBYNUMBER Exit (proto=0 Return = NULL)



Winsock Trace Examples Con't

Activate trace and redirect output to the file, trace.out. The formatter (wsformat) accepts trace input from the file, trace.out, and prints formatted output on the screen:

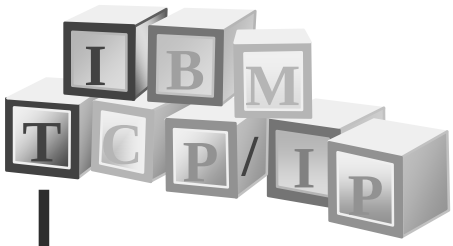
1. wstrace trace.out
2. wsformat trace.out

Activate Trace and redirect output to the default pipe, \pipe\wspipe. The formatter, which must be activated first, accepts input from the default pipe and writes the formatted output to a file, trace.out. It also creates a copy of the binary trace output in a file called trace.bin:

1. wsformat -p -f trace.out -b trace.bin (OS/2 Session A)
2. wstrace -p (OS/2 Session B)

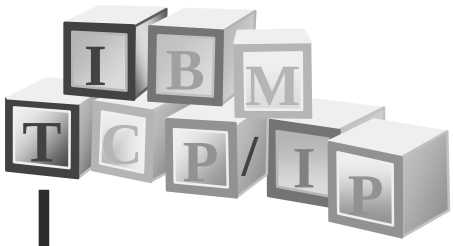
Activate trace and redirect output to COM1. The formatter, which must be activated first, accepts input from COM1 and writes the formatted output to a file, trace.out. The OS/2 MODE command is first issued to synchronize COM port settings on both machines.

1. mode com1 :14400, N,8, 1,p (OS/2 session on both machines)
   (14400 baud, no parity, 8 data bits, 1 stop bit, 30 sec. timeout)
2. wsformat com1 -f trace.out
3. wstrace on com1



Source Overview

- DAPWSOCK/PMWSOCK Module/Function List
  - ASYNC.C
    - WSAAsyncGetXbyY()s
    - WSAAsyncSelect()
    - WSACancelAsyncRequest()
    - Internal functions used to control asynchronous operation.
  - BLOKHOOK.C
    - WSAIsBlocking()
    - WSASetBlockingHook()
    - WSAUnhookBlockingHook()
    - WSACancelBlockingCall()
    - Internal functions used to control blocking hook operation.
  - BSD.C
    - BSD socket APIs
    - Internal function used to determine the socket type.
    - __WSAFDIsSet()



Source Overview Continued

- COPY.C
  - Internal functions used to copy result from the real getXbyY() to user-defined buffer.
- TRACE.C
  - Internal functions used to handle tracing.
  - panic(): an exit function when the DLL encounters "catastrophic" errors.
- TRFUNCTS.C
  - Internal trace functions
- WSA.C
  - WSAGetLastError()
  - WSASetLastError()
  - WSAStartup()
  - WSACleanup()